# 第一种关联关系：一对多（多对一）

"一对多"是最普遍的映射关系，简单来讲就如消费者与订单的关系。

**一对多**：从消费者角的度来说一个消费者可以有多个订单，即为一对多。

**多对一**：从订单的角度来说多个订单可以对应一个消费者，即为多对一。

一对多关系在hbm文件中的配置信息：

消费者（一方）：

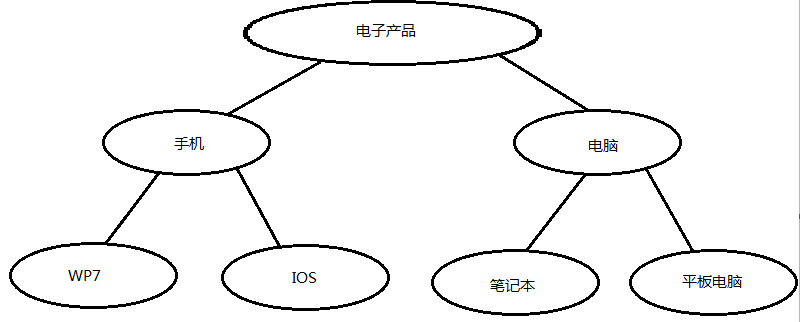
<?xml version="1.0" encoding="utf-8"?>  
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
 <hibernate-mapping>  
 <class name="com.suxiaolei.hibernate.pojos.Customer" table="customer">  
 <!-- 主键设置 -->  
 <id name="id" type="string">  
 <column name="id"></column>  
 <generator class="uuid"></generator>  
 </id>  
 <!-- 属性设置 -->  
 <property name="username" column="username" type="string"></property>  
 <property name="balance" column="balance" type="integer"></property>  
   
 <set name="orders" inverse="true" cascade="all">  
 <key column="customer\_id" ></key>  
 <one-to-many class="com.suxiaolei.hibernate.pojos.Order"/>  
 </set>  
 </class>  
 </hibernate-mapping>

订单（多方）：

<?xml version="1.0" encoding="utf-8"?>  
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
 <hibernate-mapping>  
 <class name="com.suxiaolei.hibernate.pojos.Order" table="orders">  
 <id name="id" type="string">  
 <column name="id"></column>  
 <generator class="uuid"></generator>  
 </id>  
   
 <property name="orderNumber" column="orderNumber" type="string"></property>  
 <property name="cost" column="cost" type="integer"></property>  
   
 <many-to-one name="customer" class="com.suxiaolei.hibernate.pojos.Customer"   
 column="customer\_id" cascade="save-update">  
 </many-to-one>   
 </class>  
 </hibernate-mapping>

　　"一对多"关联关系，Customer方对应多个Order方，所以Customer包含一个集合用于存储多个Order，Order包含一个Customer用于储存关联自己的Customer。

一对多关联关系有一种特例：自身一对多关联。例如：



自身一对多关联自身的hbm文件设置：

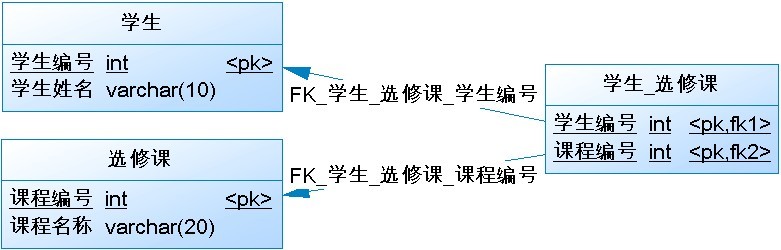
<?xml version="1.0" encoding="utf-8"?>  
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
   
 <hibernate-mapping>  
 <class name="com.suxiaolei.hibernate.pojos.Category" table="category">  
 <id name="id" type="string">  
 <column name="id"></column>  
 <generator class="uuid"></generator>  
 </id>  
   
 <property name="name" column="name" type="string"></property>  
   
 <set name="chidrenCategories" cascade="all" inverse="true">  
 <key column="category\_id"></key>  
 <one-to-many class="com.suxiaolei.hibernate.pojos.Category"/>  
 </set>  
   
 <many-to-one name="parentCategory" class="com.suxiaolei.hibernate.pojos.Category" column="category\_id">  
 </many-to-one>  
   
 </class>  
 </hibernate-mapping>

外键存放父亲的主键。

# 第二种关联关系：多对多

　　多对多关系也很常见，例如学生与选修课之间的关系，一个学生可以选择多门选修课，而每个选修课又可以被多名学生选择。**[数据库](http://lib.csdn.net/base/14" \o "MySQL知识库" \t "http://blog.csdn.net/xrt95050/article/details/_blank)**中的多对多关联关系一般需采用中间表的方式处理，将多对多转化为两个一对多。

数据表间多对多关系如下图：



多对多关系在hbm文件中的配置信息：

学生：

<?xml version="1.0" encoding="utf-8"?>  
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
<hibernate-mapping>  
 <class name="com.suxiaolei.hibernate.pojos.Student" table="student">  
 <id name="id" type="integer">  
 <column name="id"></column>  
 <generator class="increment"></generator>  
 </id>  
  
 <property name="name" column="name" type="string"></property>  
  
 <set name="courses" inverse="false" cascade="save-update" table="student\_course">  
 <key column="student\_id"></key>  
 <many-to-many class="com.suxiaolei.hibernate.pojos.Course"  
 column="course\_id"></many-to-many>  
 </set>  
 </class>  
</hibernate-mapping>

课程：

<?xml version="1.0" encoding="utf-8"?>  
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
<hibernate-mapping>  
 <class name="com.suxiaolei.hibernate.pojos.Course" table="course">  
 <id name="id" type="integer">  
 <column name="id"></column>  
 <generator class="increment"></generator>  
 </id>  
  
 <property name="name" column="name" type="string"></property>  
  
 <set name="students" inverse="true" cascade="save-update" table="student\_course">  
 <key column="course\_id"></key>  
 <many-to-many class="com.suxiaolei.hibernate.pojos.Student"  
 column="student\_id"></many-to-many>  
 </set>  
 </class>  
</hibernate-mapping>

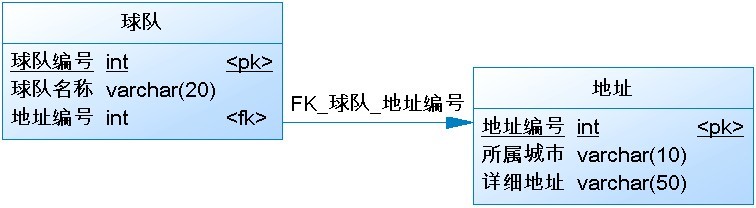
　　其实多对多就是两个一对多，它的配置没什么新奇的相对于一对多。在多对多的关系设计中，一般都会使用一个中间表将他们拆分成两个一对多。<set>标签中的"table"属性就是用于指定中间表的。中间表一般包含两个表的主键值，该表用于存储两表之间的关系。由于被拆成了两个一对多，中间表是多方，它是使用外键关联的，<key>是用于指定外键的，用于从中间表取出相应的数据。中间表每一行数据只包含了两个关系表的主键，要获取与自己关联的对象集合，还需要取出由外键所获得的记录中的另一个主键值，由它到对应的表中取出数据，填充到集合中。<many-to-many>中的"column"属性是用于指定按那一列的值获取对应的数据。

例如用course表来说，它与student表使用一个中间表student\_course关联。如果要获取course记录对应的学生记录，首先需要使用外键"course\_id"从student\_course表中取得相应的数据，然后在取得的数据中使用"student\_id"列的值，在student表中检索出相关的student数据。其实，为了便于理解，你可以在使用course表的使用就把中间表看成是student表，反之亦然。这样就可以使用一对多的思维来理解了，多方关联一方需要外键那么在本例子中就需要"course\_id"来关。

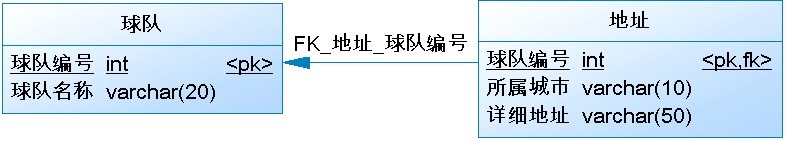
# **第三种关联关系：一对一**

　　一对一关系就球队与球队所在地之间的关系，一支球队仅有一个地址，而一个地区也仅有一支球队（貌似有点勉强，将就下吧）。数据表间一对一关系的表现有两种，一种是外键关联，一种是主键关联。图示如下：

一对一外键关联：



一对一主键关联：要求两个表的主键必须完全一致，通过两个表的主键建立关联关系：



**一对一外键关联在hbm文件中的配置信息：**

地址：

<?xml version="1.0" encoding="utf-8"?>  
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
<hibernate-mapping>  
 <class name="com.suxiaolei.hibernate.pojos.Adress" table="adress">  
 <id name="id" type="integer">  
 <column name="id"></column>  
 <generator class="increment"></generator>  
 </id>  
  
 <property name="city" column="city" type="string"></property>  
   
 <one-to-one name="team" class="com.suxiaolei.hibernate.pojos.Team" cascade="all"></one-to-one>  
  
 </class>  
</hibernate-mapping>

球队：

<?xml version="1.0" encoding="utf-8"?>  
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
<hibernate-mapping>  
 <class name="com.suxiaolei.hibernate.pojos.Team" table="team">  
 <id name="id" type="integer">  
 <column name="id"></column>  
 <generator class="increment"></generator>  
 </id>  
  
 <property name="name" column="name" type="string"></property>  
   
 <many-to-one name="adress" class="com.suxiaolei.hibernate.pojos.Adress" column="adress\_id" unique="true"></many-to-one>  
  
 </class>  
</hibernate-mapping>

　　一对一外键关联，其实可以看做是一对多的一种特殊形式，多方退化成一。多方退化成一只需要在<many-to-one>标签中设置"unique"="true"。  
**一对一主键关联在hbm文件中的配置信息：**

地址：

<hibernate-mapping>  
 <class name="com.suxiaolei.hibernate.pojos.Adress" table="adress">  
 <id name="id" type="integer">  
 <column name="id"></column>  
 <generator class="increment"></generator>  
 </id>  
  
 <property name="city" column="city" type="string"></property>  
   
 <one-to-one name="team" class="com.suxiaolei.hibernate.pojos.Team" cascade="all"></one-to-one>  
  
 </class>  
</hibernate-mapping>

球队：

<hibernate-mapping>  
 <class name="com.suxiaolei.hibernate.pojos.Team" table="team">  
 <id name="id" type="integer">  
 <column name="id"></column>  
 <generator class="foreign">  
 <param name="property">adress</param>  
 </generator>  
 </id>  
  
 <property name="name" column="name" type="string"></property>  
   
 <one-to-one name="adress" class="com.suxiaolei.hibernate.pojos.Adress" cascade="all"></one-to-one>  
  
 </class>  
</hibernate-mapping>

一对一主键关联，是让两张的主键值一样。要使两表的主键相同，只能一张表生成主键，另一张表参考主键。

<generator class="foreign">

　　<param name="property">adress</param>

</generator>

"class"="foreign"就是设置team表的主键参照adress属性的主键值。